技术雷达

Volume 30

2024 年 4 月

针对当今科技领域发展的前沿指南

/thoughtworks

Strategy. Design. Engineering.

关于技术雷达

Thoughtworker 酷爱技术。我们致力于建造技术，研究技术，测试技术，开源技术，书写技术，并不断改进技术。支持卓越软件并掀起 IT 革命是我们的使命，Thoughtworks 技术雷达就是为了完成这一使命。它由 Thoughtworks 中一群资深技术领导组成的技术顾问委员会，通过定期讨论 Thoughtworks 的全球技术战略以及对行业有重大影响的技术趋势而创建。

技术雷达以独特的形式记录技术顾问委员会的讨论结果，从首席技术官到开发人员，雷达将会为各路利益相关方提供价值。这些内容只是简要的总结。

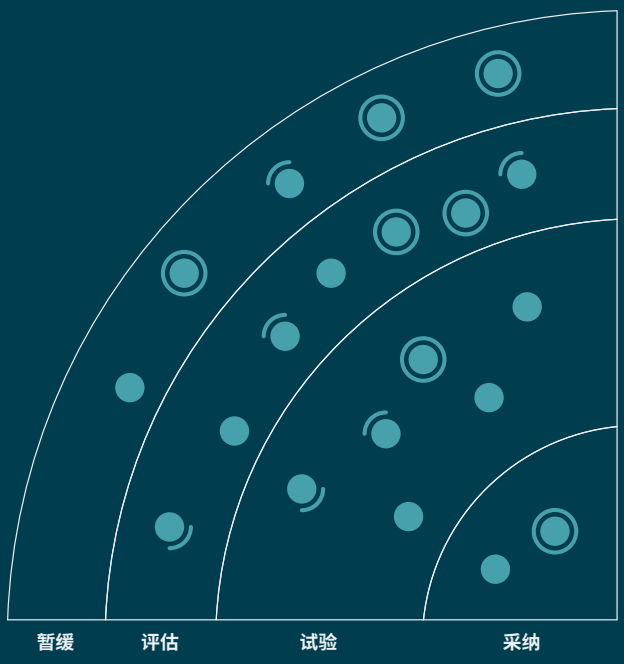
我们建议您探索雷达中提到的内容以了解更多细节。技术雷达的本质是图形性质，把各种技术项目归类为技术、工具、平台和语言和框架。如果技术可以被归类到多个象限，我们选择看起来最合适的一个。我们还进一步将这些技术分为四个环以反映我们目前对其的态度。

想要了解更多技术雷达相关信息，请点击：thoughtworks.com/cn/radar/faq

雷达一览

技术雷达持续追踪有趣的技术是如何发展的，我们将其称之为条目。在技术雷达中，我们使用象限和环对其进行分类，不同象限代表不同种类的技术，而圆环则代表我们对它作出的成熟度评估。

软件领域瞬息万变，我们追踪的技术条目也如此，因此您会发现它们在雷达中的位置也会改变。



采纳：我们强烈主张业界采用这些技术。我们会在适当时候将其用于我们的项目。

试验：值得追求。重要的是理解如何建立这种能力，企业应该在风险可控的项目中尝试此技术。

评估：为了确认它将如何影响你所在的企业，值得作一番探究。

暂缓：谨慎推行。

新的　挪进 / 挪出　没有变化

技术雷达是具有前瞻性的。为了给新的技术条目腾出空间，我们挪出了近期没有发生太多变化的技术条目，但略去某项技术并不表示我们不再关心它。

贡献者

技术顾问委员会（TAB）由 Thoughtworks 的 22 名高级技术专家组成。 TAB 每年召开两次面对面会议，每两周召开一次视频会议。其主要职责是为 Thoughtworks 的首席技术官 Rachel Laycock 和名誉首席技术官 Rebecca Parsons 提供咨询建议。

作为一个综合型组织，TAB 能够审视影响 Thoughtworks 技术战略和技术人员的各种主题。本期技术雷达内容基于 2024 年 2 月技术委员会成员在纽约的面对面会议创建。


Rebecca Parsons（CTO Emerita）


Rachel Laycock（CTO）


Martin Fowler（Chief Scientist）


Bharani Subramaniam


Birgitta Böckeler


Brandon Byars


Camilla Falconi Crispim


Erik Dörnenburg


Fausto de la Torre


Hao Xu


James Lewis


Marisa Hoenig


Maya Ormaza


Mike Mason

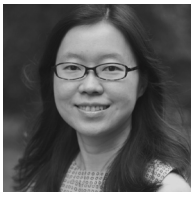

Neal Ford


Pawan Shah


Scott Shaw


Selvakumar Natesan

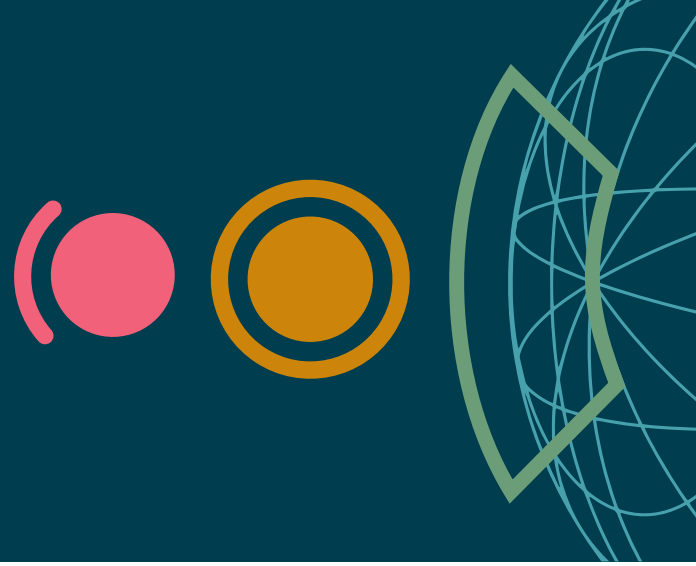

Shangqi Liu


Sofia Tania


Vanya Seth


Will Amaral

本期主题



（或许）开源的许可证

在我们的会议中，关于许可证引发了两类讨论。首先，多年来，开源软件开发生态系统依赖于一组由开源倡议组织（Open Source Initiative，OSI）编录的许可证，大多数情况下仅使用流行的几种。然而，最近发生了一些变化。一些知名工具，最近因为其维护者突然从开源模式切换到商业模式而受到了一些负面评价。当然，我们愿意为软件付费，并且认可对于额外功能使用商业许可证的常见模式。只是我们觉得，当这种转变出现在一个受众广泛的工具的核心功能上，尤其是当一个生态系统已经围绕该工具发展起来时，这是有问题的。其次，另一个有趣的转变出现在一些自诩开源的软件上，其基本功能只有在消费者支付订阅费或其他费用后才能使用。尽管这种商业模式以前就存在，但似乎在许多闪亮的新 AI 工具中被更多地利用了——它们提供了一些在细则之下过于隐藏的惊人功能。我们建议特别关注许可证问题。在使用中要提起警惕，并确保存储库中的所有文件都受到顶层许可证的覆盖。

人工智能助力软件开发团队

显然，人工智能目前正在讨论中占主导地位：技术雷达中约有三分之一的类目与之相关。我们不仅讨论了面向开发者的人工智能工具，如 GitHub Copilot，CodiumAI，aider 和 Continue，我们还探讨了如何在整个团队中全面使用 AI、以及如何利用人工智能改变软件开发的各个方面。这其中包括一系列最终未能入选的工具，例如 Warp 这样的人工智能辅助终端，将截图转换为代码的能力、由 LLMs 支持的 ChatOps 以及许多其他主题。尽管开发人员工具正在发展的日臻成熟，但我们始终对于“软件开发的所有方面都可以从人工智能和相关工具的务实使用中受益”抱有怀疑，我们正在积极跟踪相关领域的创新。与此同时，在人工智能带来近乎神奇的新技能时，相关的质量与安全风险也在涌现，这需要团队保持警惕，包括让非开发人员意识到潜在的风险。

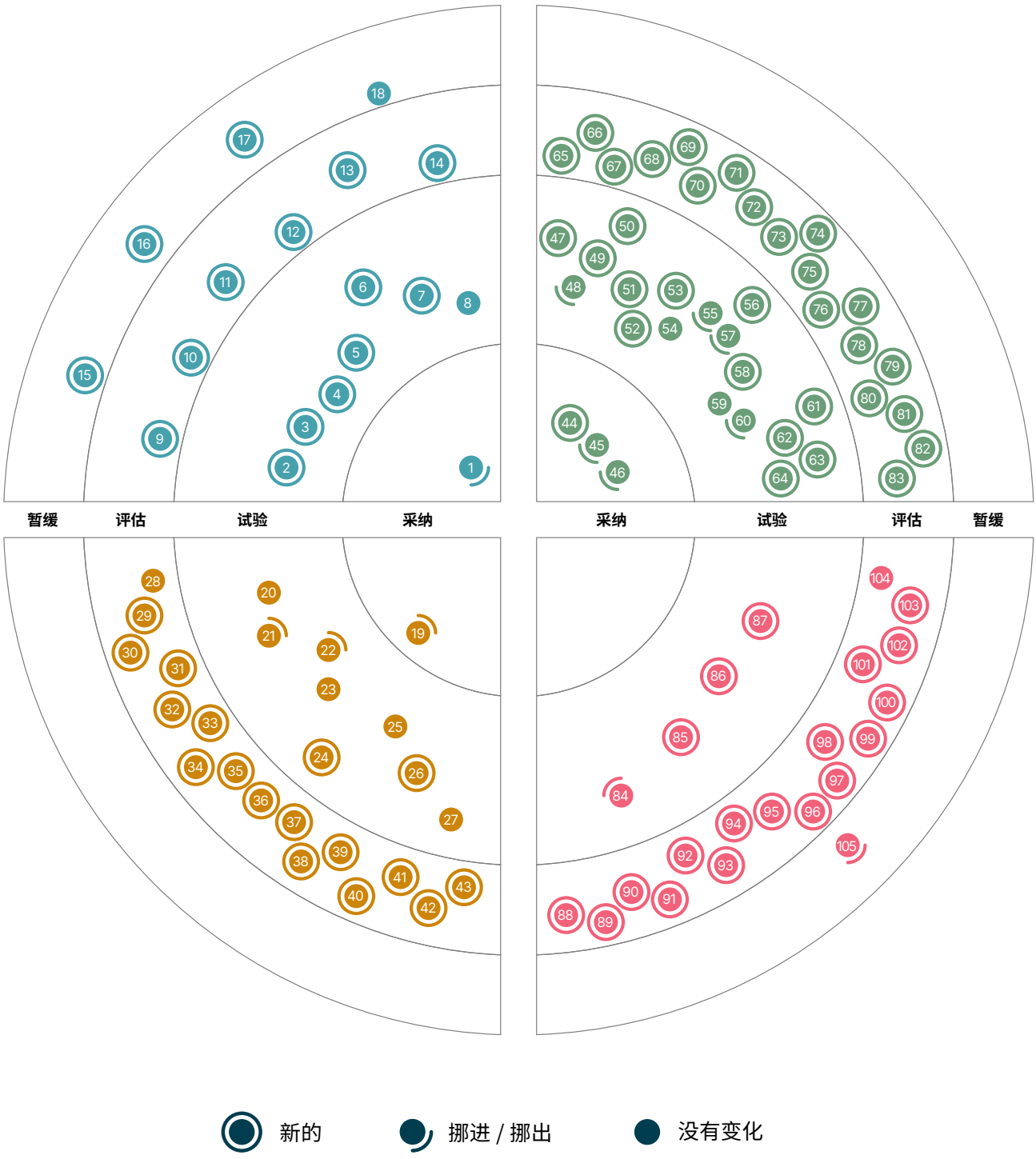
涌现的大语言架构模式

在技术领域，“模式”因为能够为特定问题提供一个简洁的解决方案名称而受到欢迎。随着大语言模型（LLMs）的日益普及，我们开始看到支持常见上下文的特定架构模式不断涌现。例如，我们讨论了 NeMo Guardrails，它允许开发人员围绕 LLM 的使用建立治理政策。我们还讨论了像 Langfuse 这样的工具，它们能够更好地观察 LLM 的输出步骤，并知道如何处理（并验证）充斥着生成代码的臃肿代码库。我们讨论了如何使用检索增强生成（RAG），这是我们偏爱的模式，以提高 LLM 输出的质量，在企业生态系统中尤其有效。此外，我们还讨论了使用低能耗（和成本）大语言模型产生材料，然后由更强大（也更划算）的大语言模型选择性审查的技术。模式为技术构建了一个重要的词汇库，随着生成式 AI 继续渗透软件开发，我们预计会看到模式（以及不可避免的反模式）的爆炸式增长。

让 Pull Request（PR）更接近正确的持续集成

Thoughtworks 一直推崇在软件开发过程中采用快速反馈循环，因此也是持续集成（CI）的大力支持者。为此，我们在 20 世纪 90 年代末构建并开源了有史以来第一个 CI 服务器 — Cruise Control 。最近，我们的首席科学家 Martin Fowler 在他的 bliki 上更新了对于持续集成的规范定义，以重申对这一实践的关注。然而，我们许多团队被迫忽视 CI/CD 中的 CI 部分，因为他们发现自己处于必须使用 Pull Request（PR）的情况。尽管 PR 的做法最初是为了管理大规模分布式的开源团队和不可靠的贡献者，然而目前已经发展成了同行评审（Peer Review，PR）的同义词，即使在紧密工作的小型团队也是如此。在这些情况下，许多开发者渴望从实践真正的 CI 中获得相同的流畅感。我们调查了几个试图减轻 PR 审查过程痛苦的工具，包括 gitStream 和 Github 合并队列。我们还讨论了诸如 stacked diffs 之类的技术，这些技术通过使集成过程更精细化，有望与 CI 的核心原则保持一致。除此之外，我们还探讨了从 PR 中提取度量的方法，以识别软件交付过程中的低效和瓶颈。工具在这个领域会起到巨大帮助，因为整体趋势是朝向生成式 AI 编程的。随着 AI 编码助手的出现，编码吞吐量增加，导致倾向于创建更大的 PR。这给异步代码审查过程增加了更大的压力。尽管我们仍然更喜欢原始的 CI 实践，但我们鼓励那些由于外部约束而无法使用 CI 的团队寻找方法，从而提高集成准确性和反馈周期速度。

本期雷达



本期雷达

技术

采纳

1. 检索增强生成（RAG）

试验

2. 自动生成 Backstage 实体描述符
3. 将传统 NLP 与 LLMs 相结合
4. 持续合规
5. 边缘函数
6. 安全标兵
7. Text to SQL
8. 追踪健康债务状况

评估

9. 人工智能团队助理
10. 对 LLM 对话进行图分析
11. 基于大语言模型的 ChatOps
12. 大语言模型驱动的自主代理
13. 使用 GenAI 理解遗留代码库
14. VISS

暂缓

15. 广泛集成测试
16. 过度热衷使用大语言模型
17. 急于冲向大语言模型微调（fine-tune LLMs）
18. 适用于 SSR 网络应用程序的 Web 组件

平台

采纳

19. CloudEvents

试验

20. 云上 Arm
21. Azure Container Apps
22. Azure OpenAI Service
23. DataHub
24. 基础设施编排平台
25. Pulumi
26. Rancher Desktop
27. Weights & Biases

评估

28. Bun
29. Chronosphere
30. DataOS
31. Dify
32. Elasticsearch Relevance Engine
33. FOCUS
34. Gemini Nano
35. HyperDX
36. IcePanel
37. Langfuse
38. Qdrant
39. RISC-V 用于嵌入式
40. Tigerbeetle
41. WebTransport
42. Zarf
43. ZITADEL

暂缓

—

工具

采纳

44. Conan
45. Kaniko
46. Karpenter

试验

47. 42Crunch API Conformance Scan
48. actions-runner-controller
49. Android 模拟器容器
50. AWS CUDOS
51. aws-nuke
52. Bruno
53. Develocity
54. GitHub Copilot
55. Gradio
56. Gradle Version Catalog
57. Maestro
58. Microsoft SBOM 工具
59. 开放策略代理（OPA）
60. Philips's self-hosted GitHub runner
61. Pop
62. Renovate
63. Terrascan
64. Velero

评估

65. aider
66. Akvorado
67. 百川 2
68. Cargo Lambda
69. Codium AI
70. Continue
71. Fern Docs
72. Granted
73. LinearB
74. LLaVA
75. Marimo
76. Mixtral
77. NeMo Guardrails
78. Ollama
79. OpenTofu
80. QAnything
81. System Initiative
82. Tetragon
83. Winglang

暂缓

—

语言和框架

采纳

—

试验

84. Astro
85. DataComPy
86. Pinia
87. Ray

评估

88. 安卓适应性
89. Concrete ML
90. Crabviz
91. Crux
92. Databricks Asset Bundles
93. Electric
94. LiteLLM
95. LLaMA-Factory
96. MLX
97. Mojo
98. Otter
99. Pkl
100. Rust for UI
101. vLLM
102. Voyager
103. WGPU
104. Zig

暂缓

105. LangChain

技术

采纳

1. 检索增强生成（RAG）

试验

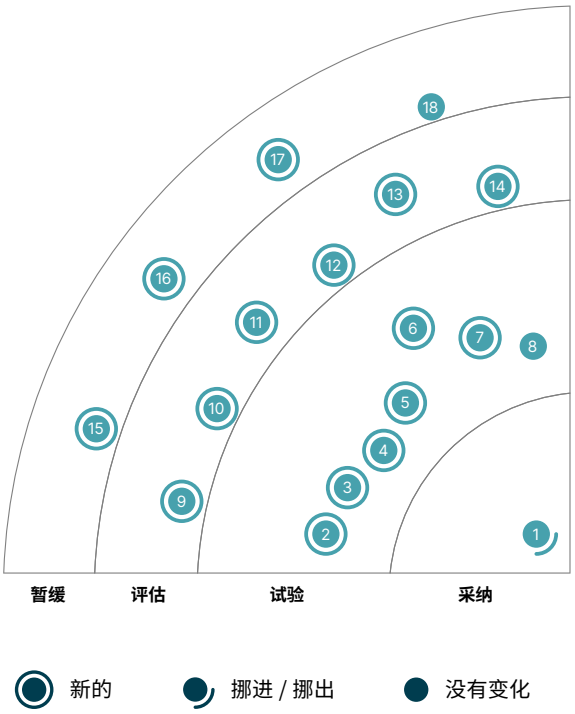
2. 自动生成 Backstage 实体描述符
3. 将传统 NLP 与 LLMs 相结合
4. 持续合规
5. 边缘函数
6. 安全标兵
7. Text to SQL
8. 追踪健康债务状况

评估

9. 人工智能团队助理
10. 对 LLM 对话进行图分析
11. 基于大语言模型的 ChatOps
12. 大语言模型驱动的自主代理
13. 使用 GenAI 理解遗留代码库
14. VISS

暂缓

15. 广泛集成测试
16. 过度热衷使用大语言模型
17. 急于冲向大语言模型微调（fine-tune LLMs）
18. 适用于 SSR 网络应用程序的 Web 组件



1. 检索增强生成（RAG）

采纳

检索增强生成（Retrieval-augmented generation，RAG）是我们团队提高大语言模型（LLM）生成响应质量的首选模式。我们已经在包括 Jugalbandi AI Platform 在内的多个项目中成功使用了它。通过 RAG，相关且可信的文档（如 HTML 和 PDF 格式）的信息被存储在支持向量数据类型或高效文档搜索的数据库中，例如 pgvector、Qdrant 或 Elasticsearch Relevance Engine。在收到给定提示后，数据库会被调取以检索相关文档，然后这些文档会与提示结合在一起，为 LLM 提供更丰富的上下文。这样一来输出质量更高，且大大减少了幻觉现象。上下文窗口——决定了 LLM 输入的尺寸是有限的，这意味着需要选择最相关的文档。我们会通过重新排序来提升提示内容的相关性。文档如果太大而无法计算嵌入，这意味着它们必须被分割成更小的块。这通常是一个困难的问题，其中一种方法是让这些块在某种程度上重叠。

2. 自动生成 Backstage 实体描述符

试验

Spotify 推出的 Backstage 已成为我们客户托管开发者体验门户的首选平台。本身来说，Backstage 只是一个托管插件，在托管的同时提供管理构成平台生态系统资产目录的界面的 shell。任何由 Backstage 显示或管理的实体都在 catalog-info 文件中配置，其中包含状态、生命周期、依赖关系和 API 等其他细节的数据。默认情况下，单个实体描述符是手动编写的，并且通常由负责相应组件的团队进行维护和版本化。保持描述符的更新可能是乏味的，并且会成为开发者采用过程中的障碍。此外，总有可能忽视变更或完全错过某些组件。我们发现 自动生成 Backstage 实体描述符 更有效且不易出错。大多数组织有现有的信息源可以启动填充目录条目的过程。良好的开发实践，例如，在 AWS 资源上放置适当的标签或向源文件添加元数据，可以简化实体发现和描述符生成。这些自动化流程可以定期运行 —— 比如每天一次 —— 以保持目录的新鲜和更新。

3. 将传统 NLP 与 LLMs 相结合

试验

大语言模型（LLMs）是自然语言处理（NLP）中的瑞士军刀。但它们往往比较昂贵，且并非总是最合适的 - 有时候使用一个螺丝刀会更合适。实际上，在 将传统 NLP 与 LLMs 相结合 ，或者在将多种 NLP 与 LLMs 相结合，以实现用例并利用 LLMs 的实际需求能力的步骤方面有很大的潜力。传统的数据科学和 NLP 方法，例如文档聚类、主题识别和分类，甚至摘要生成，成本更低且可能更有效地解决你的使用案例问题的一部分。然后，在需要生成和总结较长文本，或将多个大型文档合并时，我们使用 LLMs，以利用其较高的注意力跨度和记忆力。例如，我们已经成功地将这些技术结合使用，从一个大型单个趋势文档语料库生成关于某一领域的全面趋势报告，同时结合传统聚类方法和 LLMs 的生成能力。

4. 持续合规

试验

持续合规 是一种实践，旨在确保软件开发过程以及相关技术一直遵守行业法规和安全标准，这一过程大量依赖自动化，人工操作可能会降低开发速度并引入错误。作为替代，组织可以自动化合规检查和审计。他们可以将工具集成到软件开发流水线中，使团队能够在开发过程的早期发现并处理合规问题。将合规规则和最佳实践编码化有助于在团队间执行一致的政策和标准。它使用户能够扫描代码变更中的漏洞、强制执行编码标准以及追

踪基础设施配置变更，以确保它们满足合规要求。最后，以上内容的自动化报告简化了审计工作，并提供了清晰的合规证据。我们已经讨论过诸如发布软件物料清单（SBOMs）和应用软件供应链层级建议的技术 — 很适合在早期进行这样的尝试。这种技术的好处是多方面的。首先，自动化能够带来更安全的软件，可以在早期识别并处理漏洞，其次，随着手动任务的消除，开发速度也会加快。最后，还能够降低成本和提高一致性。对于像软件驱动汽车这样的安全关键行业，自动化持续合规可以提高认证过程的效率和可靠性，最终造就更安全、更可靠的车辆。

5. 边缘函数

试验

尽管不是一个新概念，我也注意到通过内容交付网络（CDNs）进行去中心化代码执行的可用性和使用量正在增长。诸如 Cloudflare Workers 或 Amazon CloudFront Edge Functions 这样的服务提供了一种机制，可以在靠近客户地理位置的地方执行无服务器代码片段。 边缘函数 不仅可以在边缘生成响应时提供更低的延迟，还可以在请求和响应从区域服务器出发和返回的途中，以特定位置的方式重写它们。例如，你可能会重写请求的 URL，以路由到一个特定服务器，该服务器拥有与请求正文中找到的字段相关的本地数据。这种方法最适合于短暂、快速运行的无状态处理，因为边缘的计算能力是有限的。

6. 安全标兵

试验

安全标兵 ** 指的是团队成员中对技术和非技术交付决策的安全后果持有批判性思维的人。他们会向团队领导提出这些问题和顾虑，并且对基本安全指南和要求有比较到位的理解。他们协助开发团队在软件交付的所有活动中都以安全意识进行思考，从而降低系统的整体安全风险。安全标兵不是一个单独的职位，而是分配给现有团队成员的责任，这些成员需要由安全从业者进行培训指导。通过这样的培训，安全标兵通过传播知识，并作为开发团队和安全团队之间的桥梁，提高团队的安全意识。安全标兵所做的事情中一个非常好的活动示例是威胁建模，它帮助团队从一开始就将安全风险考虑在内。在团队中任命和培训安全标兵是一个很好的开始，但仅仅依赖标兵而没有来自领导层的适当投入可能会导致问题。根据我们的经验，建立安全意识需要整个团队及管理者的投入。

7. Text to SQL

试验

Text to SQL 是一种用于将自然语言查询转换为可以由数据库执行的 SQL 查询的技术。尽管大语言模型能够理解和转换自然语言，但在你自己的 schema 中创建准确的 SQL 仍然存在很大的挑战。为此可以引入 Vanna，它是一个在 Python 中用于 SQL 生成的检索增强生成（RAG）开源框架。Vanna 分两步工作：首先你需要使用数据定义语言语句（DDLs）和示范 SQL 描述你的结构，并为它们创建嵌入向量，然后再用自然语言提出问题。尽管 Vanna 可以与任何大语言模型协作，我们还是推荐你评估 NSQL，它是一个用于 Text to SQL 任务的领域特定大语言模型。

8. 追踪健康债务状况

试验

通过将健康度评级与其他服务级目标（SLO）同等对待，并据此确定增强的优先级，而不是仅仅关注跟踪技术债务，我们不断体验到团队对其生态系统的改进。通过有效分配资源来解决与健康状况相关的最有影响的问题，团队和组织可以降低长期维护成本，更高效地发展产品。这种方法还能加强技术和非技术利益相关者之间的沟通，促进对系统状态的共同理解。尽管不同组织的衡量标准可能有所不同（请参阅本博文 中的示例），但它们最终都有助于实现长期可持续性，并确保软件保持适应性和竞争力。在瞬息万变的数字环境中，专注于 跟踪系统的健康状况与债务 ，可为维护和增强系统提供结构化的循证战略。

9. 人工智能团队助理

评估

像 GitHub Copilot 这样的 AI 编码辅助工具目前主要是在帮助和增强个人工作的背景下讨论的。然而，软件交付仍然是团队工作，并将始终是团队工作，因此你应该寻找创建 AI 团队助理 的方法来帮助创建“10 倍团队”，而不是一群孤立的 AI 辅助的 10 倍工程师。我们已经开始使用一种团队辅助方法，通过结合提示和知识源来增强知识放大、技能提升和对齐。标准化的提示有助于在团队环境中使用已经达成共识的最佳实践，例如编写用户故事的技术和模板，或实施威胁建模等实践。除了提示之外，通过检索增强生成提供的知识源，可以从组织指南或行业特定的知识库中提供与上下文相关的信息。这种方法使团队成员能够及时获得他们需要的知识和资源。

10. 对 LLM 对话进行图分析

评估

由大语言模型（LLMs）支持的聊天机器人正变得非常流行，我们看到围绕这些机器人的产品化和生产化都涌现出了许多新技术。其中一个产品化挑战是理解用户如何与这类聊天机器人展开交流，毕竟这种对话有多个发展方向。了解对话流的实际情况对于改进产品和提高转化率至关重要。有一种技术对解决这一问题大有裨益，就是 对 LLM 对话进行图分析（graph analysis for LLM-backed chats）。那些支持特定期望结果的聊天代理 — 如购物行为或成功解决客户问题 — 通常可以表示为一个期望的状态机。通过将所有对话加载到一个场景中，你可以分析它实际所处的模式，并寻找与预期状态机的偏差。这有助于发现错误和进行产品改进。

11. 基于大语言模型的 ChatOps

评估

基于大语言模型的 ChatOps** 是通过聊天平台（主要是 Slack）应用大语言模型的新兴方式，允许工程师通过自然语言来构建、部署和操作软件。这种方式有可能通过增强平台服务的可发现性和用户友好性来简化工程工作流程。截至撰写本文时，两个早期示例分别是 PromptOps 和 Kubiya。然而，考虑到生产环境需要的精细管理，组织在让这些工具接近生产环境前应该彻底评估它们。

12. 大语言模型驱动的自主代理

评估

随着像 Autogen 和 CrewAI 这样的框架的出现，LLM（大型语言模型）驱动的自主代理 正在从单一代理和静态多代理系统发展到更先进的阶段。这些框架允许用户定义具有特定角色的代理，分配任务，并使代理通过委派或对话合作完成这些任务。类似于早期出现的单一代理系统，如 AutoGPT，单个代理可以分解任务，利用预配置的工具，并请求人工输入。尽管这一领域仍处于开发的早期阶段，但它发展迅速，并且拥有广阔的探索空间。

13. 使用 GenAI 理解遗留代码库

评估

生成式人工智能（GenAI）和大语言模型（LLMs）可以帮助开发者编写和理解代码。在实际应用中，目前主要体现在较小的代码片段，但更多的产品和技术正在涌现，用于 利用 GenAI 理解遗留代码库 。这在遗留代码库文档记录不完整、或者过时的文档具有误导性时尤其有用。例如，Driver AI 或 bloop 使用了 RAG ，结合了语言智能、代码搜索与 LLMs，以帮助用户在代码库中定位自己的位置。更大的上下文窗口的新兴模型也将帮助这些技术更适配大型代码库。GenAI 对遗留代码的另一个有前景的应用是在大型机（mainframe）现代化领域，这里的瓶颈通常围绕着需要理解现有代码库、并将这种理解转化为现代化项目需求的逆向工程师。这些逆向工程师有了 GenAI 的帮助可以更快地完成工作。

14. VISS

评估

Zoom 最近开源了其漏洞影响评分系统（Vulnerability Impact Scoring System）—— VISS。这个系统主要关注的是对安全措施的漏洞评分的优先级排序。VISS 与通用漏洞评分系统（CVSS）的不同之处在于，它不侧重于对最坏情况进行预测，而是试图从防御者的角度更客观地衡量漏洞的影响。为此，VISS 提供了一个基于网页的 UI，基于多个参数来计算漏洞分数 — 这些参数按照平台、基础设施和数据组进行分类 — 包括对平台的影响、影响的租户数量、数据影响等。尽管我们对这个特定工具还没有太多的实践经验，但我们认为这种基于行业上下文的优先级定制的评估方法是值得考虑的。

15. 广泛集成测试

暂缓

当我们对自动化测试表示赞扬时，也持续看到许多组织在我们认为无效的 广泛集成测试 上投入过多。“集成测试”这个术语在表述上有些模糊不清，我们尝试引用 Martin Fowler 在该主题上的 bliki 条目描述——该分类指的是需要所有运行时依赖项的实时版本的测试。这样的测试显然是昂贵的，因为它需要具备所有必要基础设施、数据和服务的全功能测试环境。管理所有这些依赖项的正确版本需要大量的协调工作，这往往会拖慢发布周期。最后，测试本身通常是脆弱且无用的。例如，要确定测试失败是由于新代码、版本依赖性不匹配还是环境不足，而错误信息很少有助于挖掘问题源头。当然，这些批评并不意味着我们认为自动化的“黑盒”集成测试普遍存在问题，但我们的确发现了一种更有帮助的方法——就是平衡对信心的需求与发布频率。可以先假设对运行时

依赖项的一组响应来验证被测试系统的行为，然后验证这些假设的两个阶段来完成。第一阶段使用服务虚拟化来创建运行时依赖项的测试替身，并验证被测试系统的行为。这简化了测试数据管理问题，并允许进行确定性测试。第二阶段可以使用契约测试来验证这些环境假设与真实依赖项。

16. 过度热衷使用大语言模型

暂缓

在急于利用最新人工智能技术的过程中，许多组织正在试图将大语言模型（LLMs）应用于各种应用，从内容生成到复杂的决策过程。LLMs 的吸引力不可否认；它们提供了看似毫不费力的解决方案来处理复杂问题，开发人员通常可以快速创建此类解决方案，而无需多年深入的机器学习经验。当 LLM-based 的解决方案多少能够工作时，就迅速部署并转向下一个任务，这可能颇具诱惑力。尽管这些基于 LLM 的价值证明是有用的，但我们建议团队仔细考虑所使用的技术以及是否 LLM 真的是正确的最终阶段解决方案。许多 LLM 可以解决的问题——如情感分析或内容分类——传统的自然语言处理（NLP）可以更便宜、更容易地解决。分析 LLM 的作用，然后评估其他潜在解决方案，不仅可以减轻 过度热衷使用大语言模型 的风险，还可以促进对人工智能技术的更细致理解和应用。

17. 急于冲向大语言模型微调（fine-tune LLMs）

暂缓

许多组织都在试图将大语言模型（LLMs）应用于他们的产品、领域或组织知识，我们看到了太多 急于冲向大语言模型微调（fine-tune LLMs）的情况。虽然这种操作的确可以强大到对特定任务的用例进行优化，但在许多情况下对大语言模型进行微调并不是必需的。最常见误用是为了让 LLM 应用程序了解特定的知识、事实或组织的代码库进行微调。在绝大多数场景下，使用检索增强生成（RAG）可以提供更好的解决方案和更优的投入产出比。微调需要大量的计算资源和专家能力，并且比 RAG 面临更多敏感和专有数据挑战。此外当你没有足够的数据进行微调时，还有欠拟合（underfitting）的风险。又或者，当你拥有太多数据时（这倒不太常见），出现过拟合（overfitting）风险。总之达到你所需要任务专业性的正确平衡是比较困难的。在你急于为应用场景进行大语言模型微调前，需要仔细考虑这些权衡和替代方案。

18. 适用于 SSR 网络应用程序的 Web 组件

暂缓

随着 Next.js 和 htmx 等框架逐步被采纳，我们看到服务端渲染（SSR）的使用越来越多。作为一种浏览器技术，在服务器上使用 web 组件并非易事。为了简化这一过程，许多框架应运而生，有时甚至使用浏览器引擎来执行操作，但复杂性依然存在。我们的开发人员发现他们需要绕过一些障碍并付出额外努力来整合前端组件和服务端组件。比开发者体验更糟糕的是用户体验：当自定义 web 组件需要在浏览器中加载和填充时，页面加载性能会受到影响，即使使用预渲染和谨慎调整组件，未样式化内容的闪现或一些布局移动几乎不可避免。正如我们在上一期技术雷达中提到的，我们的一个团队因为这些问题不得不将他们的设计系统从基于 web 组件的 Stencil 迁移出去。最近，我们从另一个团队收到报告，他们最终用浏览器端组件替换了服务器端生成的组件，其原因是开发的复杂性。我们建议谨慎使用 用于 SSR web 应用的 web 组件 ，即使框架支持。

平台

采纳

19. CloudEvents

试验

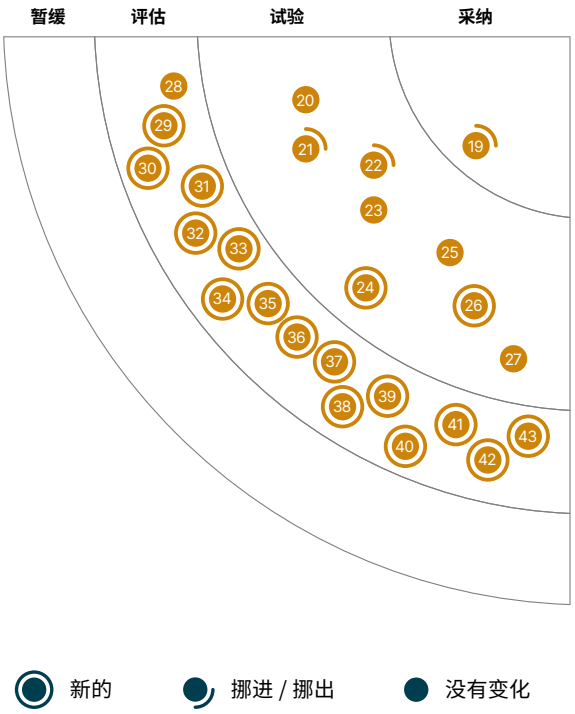
20. 云上 Arm
21. Azure Container Apps
22. Azure OpenAI Service
23. DataHub
24. 基础设施编排平台
25. Pulumi
26. Rancher Desktop
27. Weights & Biases

评估

28. Bun
29. Chronosphere
30. DataOS
31. Dify
32. Elasticsearch Relevance Engine
33. FOCUS
34. Gemini Nano
35. HyperDX
36. IcePanel
37. Langfuse
38. Qdrant
39. RISC-V 用于嵌入式
40. Tigerbeetle
41. WebTransport
42. Zarf
43. ZITADEL

暂缓

—



19. CloudEvents

采纳

事件是事件驱动架构或无服务器应用中的常见机制。然而，生产者或云提供商对它们的支持形式却存在很大差异，这阻碍了跨平台和基础设施的互操作性。CloudEvents 是一种规范，用于以通用格式描述事件数据，以实现跨服务、平台和系统的互操作性。它提供了多种语言的 SDK，因此你可以将该规范嵌入到你的应用程序或工具链中。我们的团队不仅将其用于跨云平台，还用于领域事件规范以及其他场景。CloudEvents 由云原生计算基金会（CNCF）托管，现已成为一个毕业项目。我们的团队默认使用 CloudEvents 构建事件驱动架构，因此我们正将其移入采纳状态。

20. 云上 Arm

试验

近年来，由于与传统基于 x86 的实例相比更具有成本和能源效率优势，云上的 Arm 计算实例变得越来越受欢迎。许多云服务提供商现在都提供基于 Arm 的实例，包括 AWS、Azure 和 GCP。 云上 Arm 的成本优势对于运行大型工作负载或需要扩展的企业特别有利。我们看到许多团队将工作负载（如 JVM 服务甚至数据库（包括 RDS））迁移到 Arm 实例，无需更改代码，构建脚本的更改也很小。新的基于云的应用程序和系统越来越默认使用云中的 Arm。根据我们的经验，我们建议所有工作负载使用 Arm 计算实例，除非存在特定于架构的依赖。支持多架构的工具，如多架构 Docker 镜像，也简化了构建和部署工作流。

21. Azure Container Apps

试验

Azure Container Apps 是一个托管的 Kubernetes 应用平台，能够简化容器化工作负载的部署。与 Azure Kubernetes Service（AKS）相比，运行容器化应用程序的操作和管理负担相对较少，但这是以牺牲一些灵活性和控制权为代价的，也是团队需要权衡的。这个领域的另一个产品，Azure Container Instances，通常不能满足生产环境的需求。我们的团队去年开始使用 Azure Container Apps，当时它还处于公开预览阶段，但那时它已经能在运行大容器时取得良好的结果。现在它已经普遍可用，我们正在考虑将其应用于更多用例。Dapr 和 KEDA Autoscaler 都是在其支持范围。

22. Azure OpenAI Service

试验

Azure OpenAI Service 通过 REST API、Python SDK 和基于 Web 的界面提供对 OpenAI 的 GPT-4、GPT-3.5-Turbo、Embeddings、DALL-E 模型等的访问。这些模型可以适用于内容生成、提取摘要、语义搜索和将自然语言转换为代码等任务。通过小样本学习（few-shot learning）和自定义超参数（hyperparameters），还可以进行微调。与 OpenAI 自己的 API 相比，Azure OpenAI 服务受益于 Azure 的企业级安全性和合规性功能，可在更多地区使用（尽管在更多地理区域可用性有限），并支持私有网络、内容过滤和手动模型版本控制。基于上述特性以及我们在具体使用中的积极体验，我们推荐已经使用 Azure 的企业考虑使用 Azure OpenAI Service，而不是 OpenAI API。

23. DataHub

试验

在使用 数据产品思维 构建产品时，数据血缘、数据可发现性、数据治理非常重要。我们的团队发现 DataHub 在这些方面能提供非常有效的支持。DataHub 的早期版本在需要更新元数据模型时，要求用户手动复制管理来自主产品的同步。近期的更新引入了通过插件实现的元数据模型定制。DataHub 的另一个有用功能是从源头到处理再到消费的强大端到端数据脉络。DataHub 既支持基于推送的集成，也支持基于拉动的数据血缘提取，可自动抓取跨数据源、调度器、协调器（通过扫描 Airflow DAG）、处理管道任务和仪表板等元数据。作为完整数据目录的一个开源选项，DataHub 逐渐成为我们团队的默认选择。

24. 基础设施编排平台

试验

组织内部的基础设施编排代码库频繁地成为维护和排查故障的时间黑洞。 基础设施编排平台 应运而生，试图将基础设施代码交付和部署工作流的各个方面变得标准化和产品化。其中包括构建一些工具，例如 Terragrunt、Terraspace；以及 IaC 工具供应商的服务，如 Terraform Cloud 和 Pulumi Cloud，以及与工具无关的平台和服务，如 env0 和 Spacelift。Terraform 特定的编排工具和服务有一个丰富的生态系统，通常被称为 TACOS（Terraform Automation and Collaboration Software），包括 Atlantis、Digger、Scalr、Terramate 和 Terrateam。每个平台支持不同的工作流程，包括 GitOps、持续交付和代码合规性。我们很期待看到这一领域涌现出更多解决方案。我们建议基础设施和平台工程团队探索如何使用这些解决方案，从而减少他们开发和维护基础设施所需的差异不大的自定义代码量。基础设施代码的结构化、共享、交付和部署的标准化也能够为测试、度量和监控基础设施的兼容工具生态系统的出现创造机会。

25. Pulumi

试验

在基础设施即代码领域，工具仍在不断进化，我们很高兴地看到 Pulumi 也不例外。该平台最近新增了对 Java 和 YAML 的支持，用于管理大规模基础设施，以及支持众多云配置和集成，使得该平台更加引人注目。对于我们的团队来说，它仍然是 Terraform 的主要替代品，用于开发多个云平台的代码。

26. Rancher Desktop

试验

Docker Desktop 许可证的变更迫使我们寻找相关替代方案，以便开发时能够在本地笔记本电脑上运行一系列容器。最近我们在使用 Rancher Desktop 上取得了不错的成效。这款免费且开源的应用程序容易下载并能够安装在苹果、Windows 或 Linux 机器上，便捷地提供了一个带有图形界面配置和监控的本地 Kubernetes 集群。虽然 Colima 已成为我们 Docker Desktop 的首选替代品，但它主要是一个 CLI 工具。相比之下，Rancher Desktop 对那些不想放弃 Docker Desktop 提供的图形界面的用户很有吸引力。像 Colima 一样，Rancher Desktop 允许你选择 dockerd 或 containerd 作为底层容器运行时。选择直接使用 containerd 可以让你摆脱对 DockerCLI 的依赖，但 dockerd 选项也提供了与其他工具的兼容性，这样可以与运行时守护进程进行通信。

27. Weights & Biases

试验

Weights & Biases 是一个机器学习（ML）平台，它通过实验跟踪、数据集版本控制、模型性能可视化和模型管理来帮助更快地构建模型。它可以集成到现有的 ML 代码中，以便将实时指标、终端日志和系统统计数据实时传输到仪表板进行进一步分析。近期，Weights & Biases 扩展到了与大语言模型可观测性相关的 Traces。Traces 可视化了提示链的执行流程以及中间的输入 / 输出，并提供了关于链执行的元数据（例如使用的 token 和开始与结束时间）。我们的团队发现它对于调试和更深入了解链式架构非常有用。

28. Bun

评估

Bun 是一个新的 JavaScript 运行时，类似于 Node.js 或 Deno。然而，与 Node.js 或 Deno 不同，Bun 是使用 WebKit 的 JavaScriptCore 而不是 Chrome 的 V8 引擎构建的。作为 Node.js 的替代品设计，Bun 是一个单一的二进制文件（用 Zig 编写），充当 JavaScript 和 TypeScript 应用程序的打包器、转译器和包管理器。自上一期技术雷达发布以来，Bun 已经从测试版发展到稳定的 1.0 版本。Bun 从头开始构建，并进行了几项优化——包括快速启动、改进的服务器端渲染和一个更快的替代包管理器——我们鼓励你评估它作为你的 JavaScript 运行时引擎。

29. Chronosphere

评估

在管理分布式架构时，考虑排序、索引和访问数据的成本与可观测性同样重要。Chronosphere 使用了独特的方法来管理成本、跟踪可观测数据的使用情况，以便组织可以考虑并权衡各种指标的成本价值。借助 Metrics Usage Analyzer，作为 Chronosphere Control Plane 的一部分，团队可以识别并排除他们很少（或从未）使用的指标，进而通过减少组织必须梳理的数据量来节省大量成本。考虑到这些优势，以及 Chronosphere 与其他云托管解决方案的可观测性工具匹配的能力，我们认为它非常值得投入使用。

30. DataOS

评估

随着越多越多人开始使用 data mesh，我们的团队一直在寻找将数据产品作为最高优先级对待的数据平台。DataOS 就是这样一款产品。它提供了从设计、构建、部署到演进数据产品的端到端生命周期管理。它提供标准化的声明式规范（用 YAML 编写），抽象了底层基础设施设置的复杂性，允许开发人员通过 CLI/API 轻松定义数据产品。它支持访问控制策略与 ABAC 以及数据策略用于过滤和脱敏数据。另一个值得注意的特性是它将数据联邦化到多种数据源的能力，这减少了数据重复和数据向中心地点的转移。DataOS 最适合于绿地场景，因为它提供了数据治理、数据可发现性、基础设施资源管理和可观测性的开箱即用解决方案。对于棕地场景，在 DataOS 外部编排资源（例如，像 Databricks 这样的数据堆栈）的能力还处于起步阶段，并且仍在不断发展中。如果你的生态系统能够接纳数据工具，DataOS 是加快你构建、部署和以端到端方式使用数据产品的不错选择。

31. Dify

评估

Dify 是一个 UI 驱动的用于开发大语言模型应用程序的平台，它使原型设计更加容易访问。它支持用户使用提示词模板开发聊天和文本生成应用。此外，Dify 支持使用导入数据集的检索增强生成（RAG），并且能够与多个模型协同工作。我们对这类应用很感兴趣。不过，从我们的使用经验来看，Dify 还没有完全准备好投入大范围使用，因为某些功能目前仍然存在缺陷或并不成熟。但目前，我们还没有发现更好的竞品。

32. Elasticsearch Relevance Engine

评估

尽管向量数据库因检索增强生成（RAG）使用案例而日益流行，但研究和经验报告表明，将传统的全文搜索与向量搜索相结合（成为混合搜索）可以生成更完善的结果。可以借助 Elasticsearch Relevance Engine（ESRE），成熟的全文搜索平台 Elasticsearch 支持了内置和自定义嵌入模型、向量搜索以及具有如倒数排序融合（Reciprocal Rank Fusion）等排名机制的混合搜索。尽管这个领域仍在发展中，但根据我们的经验，使用 ESRE 的这些功能以及 Elasticsearch 自带的传统过滤、排序和排名功能已经取得不错的结果，这表明支持语义搜索的搜索平台不应被忽视。

33. FOCUS

评估

云和 SaaS 计费数据可能非常复杂，不同供应商之间存在许多不一致。 FinOps Open Cost and Usage Specification（FOCUS）旨在通过提供包含一组专业术语的规范（和 FinOps framework 对齐）、一个模式和一组最低要求的计费数据来减少此类问题。规范旨在支持各类 FinOps 从业者常见的用例。即便它目前仍然处于早期开发和采用阶段，也是值得关注的。随着行业采用的增长，FOCUS 将使平台和终端用户更容易、更全面地了解在云和 SaaS 供应商那里的长尾支出情况

34. Gemini Nano

评估

Google 的 Gemini 是一系列基础性大语言模型，用于在从数据中心到手机等各种硬件上运行。其团队已经对 Gemini Nano 做了特别对优化和简化，以便在移动芯片加速器上运行。它可以实现高质量的文本摘要、上下文智能回复和高级语法纠正等功能。例如，Gemini Nano 的语言理解能力使得 Pixel 8 Pro 能够在录音机应用中总结内容。在设备上运行消除了许多来自云的系统相关的延迟和隐私问题，并且能够在没有网络连接的情况下工作。Android AICore 简化了将模型集成到 Android 应用中的过程，但在撰写本文时只支持少数设备。

35. HyperDX

评估

HyperDX 是一个开源的可观测性平台，整合了可观测性的三大支柱：日志、指标和追踪。在它的帮助下，可以实现端到端的关联，并且只需几次点击就能从浏览器会话回放跳转到日志和追踪信息。该平台使用 ClickHouse

作为所有遥感数据的中心数据存储，能够扩展以聚合日志模式，并将数十亿事件压缩成独特的集群。虽然这并不是唯一可选的可观测性平台，但我们在这里想特别强调 HyperDX 统一的开发者体验。

36. IcePanel

评估

IcePanel 通过使用 C4 模型，促进了协作式的架构建模和图表绘制，使技术和业务利益相关者能够根据需要深入到所需的技术细节级别。它支持建模架构对象，这些对象的元数据和连接可以在多个图表中重用，并且能够将这些对象之间的交互 可视化出来。版本控制和标签使协作者能够模拟不同的架构状态（例如，现状与未来），并跟踪架构各部分的用户定义分类。我们正在关注 IcePanel，因为它有潜力改善架构协作，特别是对于拥有复杂架构的组织。如果你正在寻找更好的支持图表即代码的替代产品，请查看 Structurizr。

37. Langfuse

评估

Langfuse 是一个用于观察、测试和监控大语言模型应用的工程平台。其 SDK 支持 Python、JavaScript 和 TypeScript，以及其他语言框架，如 OpenAI、LangChain 和 LiteLLM。用户可以自行托管开源版本，也可以将其用作付费云服务。我们的团队在使用它调试复杂的 LLM 链、分析完成情况以及跨用户、会话、地理、功能和模型版本监控关键指标（如成本和延迟）方面体验良好。如果你希望构建基于数据驱动的大语言模型应用程序，Langfuse 是一个值得考虑的好选择。

38. Qdrant

评估

Qdrant 是一个使用 Rust 实现的开源向量数据库。 在 2023 年 9 月的数据雷达中，我们提到过 pgvector，一个基于 PostgreSQL 的插件，可用于向量搜索。 然而，如果你需要在多个节点之间横向扩展向量数据库，我们建议考虑一下 Qdrant。 它内置的单指令 / 多数据（SIMD）加速支持可提升搜索表现，它还能帮助你将 JSON 格式的负载（payload）与向量关联起来。

39. RISC-V 用于嵌入式

评估

随着 Arm 架构继续扩大其影响力 — 我们在本期雷达更新了对云上 Arm 的评估 — 对更新但不那么成熟的 RISC-V 架构的兴趣也在增长。RISC-V 并没有在性能或效率方面带来突破 — 实际上，它的每瓦性能与 Arm 相似，且在绝对性能上无法与之竞争 — 但它是开源的、模块化的，且不依赖于单一公司。这使得它成为嵌入式系统中的一个有吸引力的选择，因为在这些系统中，架构的许可专有成本是一个重要的考虑因素。这也是为什么 RISC-V 用于嵌入式 领域日臻成熟，以及包括 SiFive 和 espressif 在内的几家公司正在为广泛的应用提供开发板和 SoCs。如今，能够运行 Linux 内核的微控制器和微处理器以及相应的软件栈和工具链都已可用。我们正在关注这一领域，并预计在未来几年内看到更多的采用。

40. Tigerbeetle

评估

Tigerbeetle 是一个用于财务会计的开源分布式数据库 。与其他数据库不同的是，它被设计成特定领域的状态机，以确保安全性和性能。集群中一个节点的状态通过 Viewstamped Replication 共识协议，以确定性顺序复制到其他节点。我们非常喜欢 Tigerbeetle 背后的 设计决策 ，使其实现具有严格可序列化保证的复式记账。这是一个相对较新且正在积极发展的数据库，尚未完全准备好投入生产使用

41. WebTransport

评估

WebTransport 是一种建立在 HTTP/3 之上的协议，提供了服务器与应用之间的双向通信。与其前身 WebSocket 相比，WebTransport 具有很多优势，包括更快的连接、更低的延迟，以及能够处理可靠且有序的数据流以及无序数据流（例如 UDP）。它可以在同一连接中处理多个流，而不会造成队头阻塞（head-of-line blocking），进而实现在复杂应用中更高效地通信。总体来说，WebTransport 适用于广泛的用例，包括实时网络应用、流媒体和物联网数据通信。尽管 WebTransport 还处于早期阶段——各浏览器的支持正逐渐成熟，流行的库如 http://socket.io 也增加了对 WebTransport 的支持——我们的团队目前正在评估其在实时物联网应用中的潜力。

42. Zarf

评估

Zarf 是一个用于离线和半连接 Kubernetes 环境的声明式软件包管理器。使用 Zarf，可以在连接到互联网时构建和配置应用程序；一旦创建完成，可以将其打包并发送到断开连接的环境以进行部署。作为一个独立的工具，Zarf 提供了几个有用的功能，包括自动生成软件物料清单（SBOM）、内置的 Docker 注册表，Gitea 和 K9s 仪表板，可从终端管理集群。云原生应用程序的网闸软件交付面临着挑战，Zarf 解决了其中大部分的问题。

43. ZITADEL

评估

ZITADEL 是一个开源的用户身份管理工具，也是 Keycloak 的替代品之一。它非常轻量级（用 Golang 编写），拥有灵活的部署选项，并且易于配置和管理。它还是多租户的，能够提供全面的功能以构建安全且可扩展的认证系统，特别是对于 B2B 应用程序，并且具有内置的安全功能，如多因素认证和审计追踪。通过使用 ZITADEL，开发者可以减少开发时间，增强应用程序的安全性，并为不断增长的用户基础进行扩容。如果你正在寻找一个用户友好、安全且开源的用户管理工具，ZITADEL 是一个不错的选择。

工具

采纳

44. Conan
45. Kaniko
46. Karpenter

试验

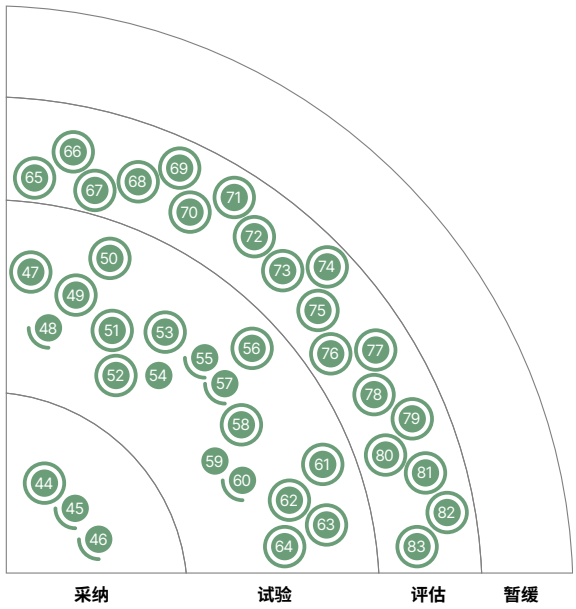
47. 42Crunch API Conformance Scan
48. actions-runner-controller
49. Android 模拟器容器
50. AWS CUDOS
51. aws-nuke
52. Bruno
53. Develocity
54. GitHub Copilot
55. Gradio
56. Gradle Version Catalog
57. Maestro
58. Microsoft SBOM 工具
59. 开放策略代理（OPA）
60. Philips's self-hosted GitHub runner
61. Pop
62. Renovate
63. Terrascan
64. Velero

评估

65. aider
66. Akvorado
67. 百川 2
68. Cargo Lambda
69. Codium AI
70. Continue
71. Fern Docs
72. Granted
73. LinearB
74. LLaVA
75. Marimo
76. Mixtral
77. NeMo Guardrails
78. Ollama
79. OpenTofu
80. QAnything
81. System Initiative
82. Tetragon
83. Winglang

暂缓

—



新的　挪进 / 挪出　没有变化

44. Conan

采纳

Conan 是一个用于 C/C++ 应用程序的开源依赖管理工具。它提供了直观的界面来定义、获取和管理依赖，使开发人员能够轻松地将第三方库集成到他们的项目中。Conan 支持所有主要操作系统，并可以适用于多种平台，包括服务器和桌面、移动和嵌入式设备。它也可以用于构建和发布 C/C++ 库和包。这些包可以通过 JFrog Artifactory 服务器在团队间共享。通过利用预构建的二进制文件，它显著节省了构建时间，特别是对于庞大的依赖。Conan 与流行的构建系统如 CMake 进行了集成，并拥有 Python SDK 用于扩展构建系统，以执行诸如签名之类的任务。根据我们的经验，Conan 改善了环境间的构建可重复性并加速了开发周期。这让代码库更加清晰易维护，对于大规模 C 和 C++ 项目而言是一个重要进步。如果你在项目中苦于依赖管理，Conan 绝对是提升开发效率的必要工具。

45. Kaniko

采纳

我们在 2022 年 10 月的技术雷达中加入了 Kaniko，就在 Kubernetes 停止支持 Docker 之后不久。当时我们强调了 Docker 作为构建基于容器流水线中容器镜像的默认工具的趋势。从那以后，我们在不同流水线的工具和配置上增加了对 Kaniko 的使用经验。我们团队欣赏它的灵活性和性能，这就是为什么我们要将其移到“采纳”阶段，以此来强调 Kaniko 应当成为这一领域的默认工具。

46. Karpenter

采纳

Kubernetes 的一个基本功能是水平自动伸缩：它能够在需要额外容量时启动新的 pods，并在负载减少时将它们关闭。然而，这只有在需要托管 pods 的节点已经存在时才有效。Cluster Autoscaler 可以做一些基本的集群扩展，由 pod 故障触发，但它的灵活性有限；然而，Karpenter 是一个更智能的、开源的 Kubernetes Operator 节点自动伸缩器：它分析当前工作负载和 pod 调度约束，选择适当的实例类型，然后根据需要启动或停止它。Karpenter 是一个像 Crossplane 这样的工具，它可以在集群外部配置云资源的操作者。尽管 Karpenter 最初是由 AWS 为 EKS 开发的，但它正在成为云 Kubernetes 服务提供商中默认的节点自动配置器，而 Azure 最近开始支持 Karpenter，推出了 AKS Karpenter Provider。

47. 42Crunch API Conformance Scan

试验

42Crunch API Conformance Scan 是一个动态测试工具，用于识别 API 文档中记录的行为与其实际实现之间的差异。此工具使用 OpenAPI 格式的 API 规格定义，概述了预期的功能和响应，并将其与 API 的实际行为进行比较。通过生成真实流量并与现场端点交互，该工具能够识别 API 承诺与其实际提供之间的任何差异。这为开发团队带来了很多好处。例如，它能在开发早期捕捉到不一致性，节省时间并防止问题进入生产环境。该工具还通过识别可能由于偏离记录行为而产生的潜在漏洞，帮助提高 API 的质量和安全性。总的来说，API Scan 能够通过识别诸如弱认证协议、不安全的数据处理实践和不充分的输入验证等问题，帮助评估 API 的安全态势。它能提供详细的报告，突出显示发现的问题以及修复建议。

48. actions-runner-controller

试验

actions-runner-controller 是一个 Kubernetes 控制器，用于操作 GitHub Actions 的自托管运行器。在需要访问 GitHub cloud runner 无法访问的资源，或具有与 GitHub 提供的不同的特定操作系统和环境要求的情况下，自托管运行器非常有帮助。这时如果团队使用 Kubernetes 集群，actions-runner-controller 将协调和扩展这些 runner。我们的团队非常赞赏它能够根据给定仓库、组织、企业或 Kubernetes 集群中运行的工作流数量来扩展 runner 的能力，以及它可以很好地处理 Linux 和 Windows runner。

49. Android 模拟器容器

试验

Android Emulator Container 通过消除因操作系统兼容性问题和系统依赖，以及设置多个 Android 版本的模拟器所带来的复杂性，使得 Android 应用测试变得更为简便。传统上，这些复杂性会带来额外的工作量致使团队完全放弃自动化测试，进而导致开发和测试周期变慢。Android 模拟器容器简化了这个过程，使其可以无缝集成到 CI 流水线中进行自动化测试。我们的团队主要使用这些容器进行设备化测试，这些测试会在每次提交时自动执行，为开发者提供即时反馈。此外，我们还使用 Android 模拟器容器来运行夜间的端到端测试。

50. AWS CUDOS

试验

我们一如既往地认为应当将成本监控列为适应性函数。云服务商提供了各种监控云消费的服务，例如 AWS Cost Explorer 或 Google Cloud FinOps Hub。在 AWS 生态系统中，我们的团队使用 CUDOS（Cost and Usage Dashboards Operations Solution）来监控大型母公司下不同业务部门或法律实体在 AWS Marketplace 的消费。该仪表板提供了全面的成本和使用细节，具有资源级别的细粒度，有助于优化成本、跟踪使用目标并实现运营卓越。

51. aws-nuke

试验

aws-nuke 是一个开源工具，解决了开发和沙箱 AWS 账户中积累未使用资源导致成本效率低下的常见挑战。该工具可以识别并删除 AWS 账户或区域内所有可删除的资源，除了默认或 AWS 管理的资源，本质上是要将环境重置为第一天的状态。它还提供自定义排除策略，以确保关键资源得到保护。我们已经在成本优化的默认用例以及灾难恢复（DR）环境中成功使用了这个工具。通过自动化清理开发和沙箱环境，aws-nuke 帮助最小化不必要的资源开销。它还有助于在演习或练习后高效拆除临时 DR 基础设施。尽管稳定，aws-nuke 是一个非常具有破坏性的工具，不适用于生产环境。请始终进行试运行，以确认不会删除必要资源。

52. Bruno

试验

Bruno 是 Postman 和 Insomnia 的开源桌面替代品，用于 API 的测试、开发和调试。它将测试集合保存在本地，因此可以使用 Git 或其他版本控制工具来进行协作。目前有一些 Thoughtworks 团队正在使用 Bruno，他们喜欢它简单并且仅支持离线使用的设计。

53. Develocity

试验

Develocity（之前是 Gradle Enterprise）解决了大型软件项目长时间的构建和测试周期的痛点。它采用构建缓存和预测试选择来提升性能从而缩短开发人员在本地和 CI/CD 环境中的反馈循环。我们的平台团队发现这对于加速构建和测试、分析命令以确定工作流程中哪些部分仍需优化、识别和解决不稳定的测试以及对运行测试的硬件进行分析非常有用。

54. GitHub Copilot

试验

尽管 AI 编码辅助市场愈发壮大，GitHub Copilot 仍然是我们的首选，且被许多团队广泛使用。自上次我们介绍 GitHub Copilot 以来，最有趣的改进来自于聊天功能。例如，不再需要用注释作为提示，这样会使代码变得混乱；相反，内置聊天可以帮助提示用户，而无需撰写注释。内联聊天还可以更改代码，而不仅仅是编写新行。现在还可以通过使用 @workspace 标签，显著扩展聊天时询问有关代码问题的上下文。这使得用户可以询问有关整个代码库的问题，而不仅仅是打开的文件。你可以通过使用 Copilot Enterprise 版本进一步扩展此上下文，该版本会从你在 GitHub 上托管的所有存储库中提取上下文。最后，GitHub 已经开始将一些聊天请求路由到更强大的基于 GPT-4 的模型，并且在流行的 Jetbrains IDE 中即将推出聊天功能（尽管在撰写本文时仍处于内测阶段）。这些发布表明，这一领域的改进步伐并未减缓。如果你去年尝试过编码助手却最终放弃，我们建议你持续关注新发布的功能，并再次尝试。

55. Gradio

试验

Gradio 是一个开源的 Python 库，它能帮助机器学习（ML）模型创建基于 web 的交互式界面。ML 模型上的图形 UI 能够帮助非技术受众更好地理解输入、约束和输出。Gradio 在生成式人工智能领域获得了大量关注，因为它让生成式模型更易于尝试和使用。通常，我们只有在生产环境中真正使用过才会将一个技术放在雷达的试验环。目前，我们团队已经多次使用它并在最近的一次大型活动中，帮助客户使用 Gradio 做了一次现场演示。我们对 Gradio 在这些用例中展现的效果非常满意，因此将其移入“试验”环。

56. Gradle Version Catalog

试验

Gradle 版本目录是 Gradle 构建工具的一个有用的功能，它允许你集中管理构建文件中的依赖项。我们的团队发现它在 Android 多模块项目中特别有用。你可以为这些依赖项创建一个中央版本目录，然后在 Android Studio 的帮助下以一种类型安全的方式引用它，而不是在单独的构建文件中硬编码依赖项名称和版本号并管理升级。

57. Maestro

试验

Maestro 在测试移动应用程序中的复杂流程时非常有用。它易于学习和理解，并且可以很方便地集成到我们的开发工作流程中。Maestro 支持一系列移动平台，包括 iOS、Android、React Native 和 Flutter 应用。其声明式的 YAML 语法简化了复杂移动 UI 交互的自动化。从工具的演进来看，它的增强功能值得关注，如全面的 iOS

支持以及引入了诸如 Maestro Studio 和 Maestro Cloud 之类的工具，我们鼓励任何希望优化其移动应用程序测试流程的人尝试一下。

58. Microsoft SBOM 工具

试验

Microsoft SBOM 工具 是一个开源工具，用于生成与 SPDX 兼容的软件物料清单（SBOM）。我们之前已经提到过软件物料清单，而这个工具其变得更加容易。SBOM 工具支持多种流行的包管理器（包括 npm、pip 和 Gradle），所以可以与许多项目兼容。它非常易于使用，可以集成到现有的开发工作流中，包括与 CI/CD 流水线的集成。借此开发者获得了多重优势，其中改善软件安全是一个关键好处，因为清晰的组件视图可以帮助识别漏洞和管理风险。许可证合规性也得到增强，因为开发者可以确保遵守所有相关协议。此外，SBOM 在软件供应链中促进透明度，帮助跟踪依赖性和缓解潜在风险。如果你正在寻求简化 SBOM 生成、提升软件安全性以及控制你的软件供应链，你应该尝试一下 Microsoft SBOM 工具。

59. 开放策略代理（OPA）

试验

Open Policy Agent（OPA）是一个统一的框架和语言，用于声明、执行和控制策略。在我们团队，它已经成为定义分布式系统策略的一种常用方式，尤其是在我们需要实施变更点的合规时。OPA 使团队能够实现各种平台工程模式，例如控制部署到 Kubernetes 集群的内容、在服务网格 中跨服务强制访问控制以及实现细粒度的访问应用程序资源的 安全策略即代码。虽然 OPA 实现存在一定复杂性，但事实证明它是一种确保 DevOps 文化下的合规性的非常有价值的工具。我们还将继续关注 OPA 在运营系统之外扩展到大数据解决方案的成熟度。

60. Philips's self-hosted GitHub runner

试验

虽然 GitHub Actions runners 已经覆盖了最常见的运行时环境，并且是最早投入使用的，但有时团队还是需要管理自托管运行器，比如当组织政策只允许从组织内部的安全边界内部署到私有托管基础设施时。在这种情况下，团队可以使用 Philips's self-hosted GitHub runner，这是一个 Terraform 模块，可以在 AWS EC2 竞价实例（spot instance）上启动自定义运行器。该模块还创建了一组 Lambda 用于处理这些运行器的生命周期管理（扩展和缩减）。根据我们的经验，这个工具极大地简化了自托管 GitHub Actions runner 的配置和管理。对于使用 Kubernetes 的团队来说，另一种选择是 actions-runner-controller。

61. Pop

试验

结对编程对于我们来说是一项必不可少的实践，因为它能帮助我们提高代码质量，在团队内传播知识。虽然结对编程最好发生在线下，面对面进行，但在不可避免要进行线上结对时，我们的团队成员也为此探索了很多工具，比如 Tuple，Visual Studio Live Share，Code With Me ，和通用的聊天及会议工具。 Pop（前身为 Screen）作为这个领域最新的工具引起了我们的关注，它来源于 Screenhero 的创建者，支持多人屏幕分享、编写注释和进行高质量音频 / 视频通话。我们有些团队正在频繁使用它来进行结对编程和远程工作会议，他们对该软件体验反馈非常正向。

62. Renovate

试验

作为软件构建过程的一部分，自动监控和更新依赖项已成为整个行业的标准实践。这样一来，在开源软件包发布安全更新时，用户不必盲目猜测。多年来，Dependabot 一直是这一实践的标准工具，但 Renovate 在出现后逐渐成为许多团队的第一选择。他们发现 Renovate 更适合现代软件开发环境，其中可部署的系统不仅依赖于代码和库，还包括运行时工具、基础设施和第三方服务。除了代码之外，Renovate 还涵盖了对这些辅助工件的依赖性。我们的团队还发现 Renovate 通过配置和定制选项提供了更多的灵活性。尽管 Dependabot 仍然是一个安全的默认选择，并且能够更方便地与 GitHub 集成，但我们建议评估 Renovate，看看它是否可以进一步减轻开发人员手动维护应用生态系统安全的负担。

63. Terrascan

试验

Terrascan 是一个用于基础设施即代码（IaC）的静态代码分析器，旨在云原生基础设施部署之前检测安全漏洞和合规问题。 它支持对 Terraform，Kubernetes（JSON/YAML），Helm，AWS CloudFormation，Azure Resource Manager，Dockerfiles 和 GitHub 进行扫描。默认策略包支持所有主流的云供应商，GitHub，Docker 和 Kubernetes. 我们团队在本地使用 Terrascan 作为预提交钩子（pre-commit hook）并在 CI 流水线中集成 Terrascan 来检测 laC 漏洞和违规行为。

64. Velero

试验

Velero 是一个用于备份和恢复 Kubernetes 资源和持久卷的开源工具。它通过启用按需和计划备份，简化了灾难恢复和集群迁移。Velero 还能更精细地控制哪些资源被备份，以及相关的备份 / 恢复流程。我们很赞赏它的易用性，以及它依赖的是 Kubernetes API 而非更低层次的 etcd 等。

65. aider

评估

aider 是一款开源的 AI 辅助编码工具。与该领域的许多开源工具一样，aider 并不直接与 IDE 集成，而是以 CLI 的形式在终端启动。目前许多辅助编码工具只能读取代码，或者一次只能更改一个文件，而 aider 的有趣之处在于，它提供了一个聊天界面，并且有写权限来对多个文件的代码库进行访问。这使得 aider 可以帮助实现跨越多个文件的概念（例如，在 HTML 中添加定位器，然后在功能测试中使用它们”），并在代码库中创建新的文件和文件夹结构（例如，创建一个与 X 文件夹中的组件类似的新组件）。由于 aider 是开源而非托管产品，因此需要提供 OpenAI 或 Azure OpenAI API 密钥才能使用。一方面，因为是按使用量计费，这非常适合轻度使用；但另一方面，aider 在与 AI API 交互时似乎很“健谈”，因此使用时要注意请求开销和费用限制。

66. Akvorado

评估

Akvorado 是一个开源的网络监控和分析工具。它可以捕获网络流量，支持 Netflow/IPFIX 和 sFlow 协议，为其添加接口名称和地理信息，然后将更新后的流量保存在 ClickHouse 中供未来分析使用。虽然 OpenTelemetry

在观测应用层流量方面越来越受欢迎，但我们通常会遇到在网络层面很难发现和解决的挑战。在这种情况下，像 Akvorado 这样的工具非常有用，它可以帮助你分析网络拓扑中各种设备之间的网络流量。

67. 百川 2

评估

百川 2 是新一代开源大型语言模型。 它采用了 2.6 万亿 Tokens 的高质量语料进行训练，在中文、英文和多语言基准测试上表现出色。 值得注意的是百川在医疗和法律等领域特定的数据集进行了针对性训练，这让我们在相关垂直领域会优先考虑它。

68. Cargo Lambda

评估

Rust 的高效和性能很适合用于无服务器计算，再考虑到它还有函数无需运行时的优势，这些都使得它能快速启动工作。但是使用 Rust 开发 lambda 函数的体验此前并不好，而这一情况被 Cargo Lambda 的出现改变了。作为 cargo 的子命令，它能集成到典型的 Rust 工作流中并且能允许你在开发机上运行和测试 AWS Lambda 函数而无需借助 Docker、虚拟机，或者其他工具。 使用 Zig 工具链，Cargo Lambda 可以在多种不同操作系统中的 Linux 沙盒中交叉编译并用于 AWS Lambda，这套工具链支持 ARM 和 Intel 作为目标架构。

69. Codium AI

评估

在 AI 编程助手持续涌现的过程中，相较于孵化大而全的工具，一些产品选择聚焦于某些领域。Codium AI 正是如此，它聚焦于使用 AI 生成代码测试。可以用于所有编程语言，但是对常用技术栈如 JavaScript 和 Python 提供了更进阶的支持。 我们格外中意这个工具，因为它不只向开发者提供测试代码，同时还能对场景提供用于评审的自然语言描述。这能使开发者理解测试用例背后的用意，帮助开发者选择哪些要放入代码库。如果想要进一步提高对特定代码库和测试用例生成的测试质量，用户可以提供测试示例和一些提示，帮助 AI 获取更多高质量的信息。

70. Continue

评估

Continue 是一种用于 VS Code 和 JetBrains IDEs 的开源 AutoPilot 工具。我们非常喜欢它，因为它通过与 IDE 的直接集成，消除了从聊天界面复制 / 粘贴到大型语言模型的痛苦。它支持多个商业和开源 模型，并且它能够帮助尝试不同的大语言模型 提供商，包括 自托管的大语言模型 。甚至可以在 没有网络连接 时运行 Continue。

71. Fern Docs

评估

目前被广泛使用的 REST APIs 的一个特点是它们的合约会被彻底地记录在案。开发者更倾向于采纳和使用那些以结构化、有组织的方式准确描述其行为和语法的 API。随着合约的发展，保持这些文档的更新可能既耗时又

容易被忽视。Fern Docs 通过减少编写和维护 API 文档的劳动强度来帮助解决这个问题。Fern 可以自动从规范文件生成一个具有吸引力、可用文档的网站，这个文件能与 API 代码一起版本化。我们对这个产品的初步印象是积极的，不过 Fern 要求用户在一个专有配置文件中维护 API 信息，尽管它提供了一种将 OpenAPI 规范转换为其自己的配置格式的方法，但我们更期待一个能够直接从带注释的源代码生成文档的工具。

72. Granted

评估

鉴于组织在 AWS 环境中常常采用多账户策略，工程师经常需要在短时间内切换多个账户。而 Granted 作为一个命令行工具，简化了同时在浏览器中打开多个账户的操作，从而使账户切换变得更加流畅。它利用浏览器的原生功能来隔离多个身份，例如 Firefox 的 多账户容器 和 Chromium 的配置文件。如果指定了特定服务（如 S3）作为参数，Granted 会打开该服务的登录页面。目前，Granted 仅支持 AWS。值得注意的是，它将 AWS SSO 的临时凭证安全地存储在密钥链中，而不是以明文形式存储在磁盘上。

73. LinearB

评估

LinearB 是一个旨在为工程领导者提供数据驱动洞察以实现持续改进的平台。它主要在三个关键领域发挥作用：基准测试、工作流自动化和投资。我们在使用 LinearB 的度量工具时发现，它有助于支持持续改进。我们的一个团队利用该平台跟踪工程度量、识别和讨论改进机会，并根据数据定义可行的步骤，从而取得了可衡量的进展。我们很高兴看到这与 LinearB 的核心价值主张相一致：基准、自动化和改进。LinearB 与 GitHub、GitLab、Bitbucket 和 Jira 集成。它提供了一套全面的预配置工程度量，重点关注 DORA 度量（部署频率、交付周期、变更失败率和恢复时间）。作为 DORA 研究定义的四个关键度量的坚定支持者，我们欣赏 LinearB 在度量对软件交付效能真正重要的维度的重视。一直以来，收集 DORA 特定度量都是一个挑战。许多团队不得不借助复杂的 CD 管道仪器、自定义仪表板或依赖手动流程。尽管我们的体验仅限于一个团队，但 LinearB 似乎是工程度量收集和跟踪以及培养数据驱动持续改进方法的一个引人注目的替代品。

74. LLaVA

评估

LLaVA（Large Language and Vision Assistant）是一个开源的大型多模态模型，它结合了视觉编码器和大语言模型，用于通用视觉和语言理解。LLaVA 在遵循指令方面的强大能力，使其成为多模态人工智能模型中的有力竞争者。最新版本，LLaVA-NeXT，能进一步提升问答能力。在开源的语言和视觉辅助模型中，与 GPT-4 Vision 相比，LLaVA 是一个很有前景的选择。我们的团队一直在使用它进行视觉问题解答。

75. Marimo

评估

Marimo 通过优先考虑复用性和交互性，为 python notebook 提供了新的体验。它解决了传统 notebook 如 Jupyter 中隐藏状态（hidden state）的挑战，该问题可能导致不可预料的结果并阻碍可复用性。Marimo 通过

将 notbook 存储为无隐藏状态的纯 python 文件和基于依赖关系（当变量改变时，所有受影响的 cell 会自动运行）的确定性执行顺序来解决这个问题。Marimo 还使用了类似的可以将 cell 值的改变传递给依赖于该 cell 的交互式 UI 元。它可以被部署为网页 app，也可以用于展示成果和进行原型设计。虽然我们对 Marimo 的潜力，尤其是数据探索和分析目标上的再现性潜力感到兴奋，我们依然谨慎对待生产化的笔记本。

76. Mixtral

评估

Mixtral 是 Mistral 发布的开放权重大语言模型家族的一部分，它采用了稀疏混合专家架构。这个模型家族以 7B 和 8x7B 参数大小的形式，提供原始预训练和微调版本。其大小、开放权重特性、基准测试中的性能以及 32,000 个 token 的上下文长度，使其成为自托管大语言模型中一个非常耀眼的选择。需要注意的是，这些开放权重模型并没有针对安全性进行优化调整，用户需要根据自己的用例进行精细调整。我们在开发与特定印度法律任务相关的数据上训练的精调 Mistral 7B 模型 Aalap 方面有一定经验，该模型在有限成本的基础上表现相当好。

77. NeMo Guardrails

评估

NeMo Guardrails 是 NVIDIA 的一个易用开源的工具包，它可以使开发人员在会话应用的大语言模型上实现一套防护措施。尽管大语言模型在构建交互式体验上有巨大的潜力，但他们在事实准确性、偏见和潜在的滥用方面上存在一些固有的局限性，这使得我们需要采取一些必要的保护措施。Guardrails 提供了一个有前景的方法来确保大语言模型的责任性和可信性。尽管当谈到大语言模型的保护措施时都会有多种选择，但是我们团队发现 NeMo Guardrails 尤其有用，因为它支持可编程的规则和运行时的集成，并且可以应用到现有的大语言模型的应用上，而不需要大量的代码修改。

78. Ollama

评估

Ollama 是一个在本机上运行并管理大语言模型的工具。 我们之前讨论过自托管大语言模型我们很高兴这个生态逐渐成熟，产生了像 Ollama 的工具。Ollama 支持多种 流行的模型 的下载和本地运行——包括 LLaMA-2，CodeLLaMA，Falcon 和 Mistral。一经下载，你可以通过命令行、接口或者开发组件与模型交互执行任务。我们正在评估 Ollama，目前看起来不错，能通过在本机运行大语言模型提升开发者体验。

79. OpenTofu

评估

OpenTofu 是对 Terraform 的一个分支，作为对 HashiCorp 最近模糊不清的许可证更改的回应。它是开源的，并已被 Linux Foundation 接受。它得到了包括第三方供应商在内的几个组织的支持。当前版本与 Terraform 的最后一个开源版本兼容。1.7 版新增了客户端加密功能。OpenTofu 项目的未来尚不明确，包括它将如何紧密支持 Terraform 未来版本的兼容性，它能否得到当前支持者的长期支持也是一个未知项。我们建议持续关注它，但要谨慎使用。如果你们团队有非常强的风险管理能力，可以尝试使用，或许还能为代码库作出一些贡献。

80. QAnything

评估

大语言模型（LLMs）和检索增强生成（RAG）技术极大地提高了我们整合和提取信息的能力。我们看到越来越多的工具利用了这一点，QAnything 就是其中之一。QAnything 是一个问答界面的知识管理引擎，能够从包括 PDF、DOCX、PPTX、XLSX 和 MD 文件等在内的多种文件格式中总结和提取信息。出于数据安全考虑，QAnything 还支持离线安装。我们的一些团队使用 QAnything 来构建他们的团队知识库。在具有更深行业深度的 GenAI 场景中（例如为投资报告生成摘要），我们也尝试使用这个工具进行概念验证，以在构建真正的产品之前展示 LLMs 和 RAG 的潜力。

81. System Initiative

评估

近年来，几乎没有新兴工具能挑战 Terraform 作为基础设施即代码工具的主导地位。尽管出现了 Pulumi、CDK 以及最近的 Wing 等替代方案，但 Terraform 的模块化、声明式范式已被证明是最持久流传的。事实上，所有这些方法都有一个共同的目标，即模块化代码创建单一的基础设施。System Initiative 是一款全新的、实验性的工具，代表了 DevOps 工作的一个新方向。可以将 System Initiative 视为基础设施的数字孪生。对 System Initiative 状态的交互式更改会导致相应的变更集，可以应用于基础设施本身。同样，基础设施的变更也会反映在 System Initiative 状态中。这种方法的一个巨大优势体现在，它为应用部署和可观测性等事务创造的协作环境。工程师通过具有整个环境图形表示的用户界面，与 System Initiative 进行交互。除了管理云基础设施，您还可以使用该工具来管理容器、脚本、工具等。尽管我们通常对这类 GUI 工具持怀疑态度，但 System Initiative 可以通过 TypeScript 代码扩展来处理新资产或执行策略。我们非常喜欢这个工具中所体现的创造性思维，并希望它能鼓励其他人突破基础设施即代码方法的现状。System Initiative 是免费和开源的，基于 Apache 2.0 许可，目前处于公开测试阶段。维护者自己还不建议将该工具用于生产环境，但我们认为它值得在当前状态下检查，体验一种完全不同的 DevOps 工具方法。

82. Tetragon

评估

Tetragon 是一个基于 eBPF 的开源安全可观测性和运行时强制执行工具。我们一段时间前在 Radar 中提到了 Falco，用于检测安全威胁。Tetragon 通过利用 eBPF 在 Linux 内核中在运行时 执行安全策略，实现了不仅限于威胁检测的能力。你可以将 Tetragon 作为一个独立工具在裸机环境中使用，也可以在 Kubernetes 环境中使用。

83. Winglang

评估

我们正在基础设施即代码（IaC）领域看到许多新动态，像 Winglang 这样的工具正在出现。Winglang 采取了一种不同的方法来定义基础设施和运行时行为。它提供了高级抽象，覆盖了由 CloudFormation、Terraform、Pulumi 和 Kubernetes 等工具提供的平台特定功能。使用 Winglang，你可以编写在编译时运行的代码以生成基础设施配置，然后编写在运行时执行的代码以进行应用程序行为。它提供了一种在本地运行的模拟模式，并集成了测试框架。我们正在关注这个有趣的工具，它可能预示了 IaC 未来发展方向。

语言和框架

采纳

—

试验

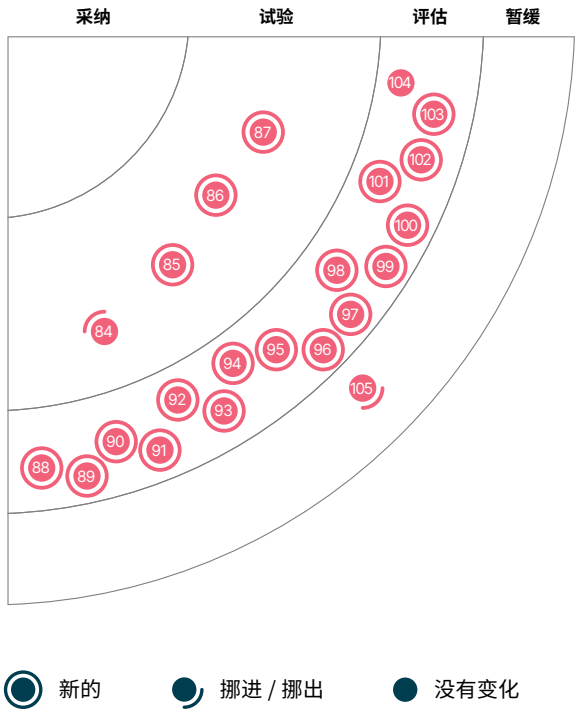
84. Astro
85. DataComPy
86. Pinia
87. Ray

评估

88. 安卓适应性
89. Concrete ML
90. Crabviz
91. Crux
92. Databricks Asset Bundles
93. Electric
94. LiteLLM
95. LLaMA-Factory
96. MLX
97. Mojo
98. Otter
99. Pkl
100. Rust for UI
101. vLLM
102. Voyager
103. WGPU
104. Zig

暂缓

105. LangChain



84. Astro

试验

Astro 框架在社区中越来越受欢迎。我们的团队已经使用 Astro 构建了诸如博客和营销网站之类的内容驱动型网站。Astro 是一个多页面应用程序框架，它在服务器上渲染 HTML 并最小化通过网络发送的 JavaScript 。我们认为不错的一点是，尽管 Astro 鼓励仅发送 HTML，但它支持在适当的情况下选择性地使用您选择的前端 JavaScript 框架编写的活动组件。它通过其岛屿架构实现这一点。岛屿是单页内的交互区域，在这里所需的 JavaScript 仅在需要时才被下载。通过这种方式，网站的大部分区域被转换为快速的静态 HTML，而 JavaScript 部分则优化为并行加载。我们的团队既喜欢其页面渲染性能，也喜欢其构建速度。Astro 组件语法是 HTML 的简单扩展，学习曲线相当平缓。

85. DataComPy

试验

对数据帧进行比较是数据工程中的常见任务，常用于确保两个数据转换方法间没有显著的偏差或不一致。DataComPy 是一个用于比较 pandas，Spark 或其他格式 DataFrame 的工具。这个库不仅能比较 DataFrame 的一致性，还能在行和列上对不一致的地方给出细致的洞见。 DataComPy 还可以指定忽略掉无需展示的数值绝对 / 相对比较和已知差异。我们的一些团队将它用于了冒烟测试组件中，他们发现它能高效比对大数据量多字段的 DataFrame，与此同时它给出的报告也易于理解和应对。

86. Pinia

试验

Pinia 是一个在 Vue.js 中使用的存储库和状态管理框架。它使用声明性语法并提供了自己的状态管理 API。与 Vuex 相比，Pinia 提供了一个更简单的 API，简化了使用流程，提供了组合式 API，并且在与 TypeScript 一起使用时具有可靠的类型推断支持。Vue.js 团队认可 Pinia 作为 Vuex 的可靠替代品，目前它是 Vue.js 的官方状态管理库。总的来说，我们认为 Pinia 的简单性和易用性是非常不错的。

87. Ray

试验

机器学习（ML）的工作负载正在变得越来越计算密集型。尽管用笔记本电脑开发训练模型很便利，但这样的单节点开发环境很难适应扩展需求。Ray AI 和 Python 代码从笔记本电脑扩展到集群的统一框架。它本质上是一个封装良好的分布式计算框架，集成了一系列 AI 库以简化 ML 的工作。通过与其他框架（例如，PyTorch 和 TensorFlow 的集成，它可以用于构建大规模 ML 平台。像 OpenAI 和字节跳动这样的公司大量使用 Ray 进行模型训练和推理。我们还使用它的 AI 库帮助我们的项目进行分布式训练和超参数调优。我们推荐你在构建可扩展的 ML 项目时尝试使用 Ray。

88. 安卓适应性

评估

有些移动应用或者游戏非常消耗资源，他们可以在几分钟内导致散热过载。在这种情况下，设备会降低 CPU 和 GPU 的频率以降低温度，但游戏帧率也会随之降低。当散热情况有所改善时，设备的帧率会回升。如此循环往

复，软件会变得非常不稳定。Android Adaptability 是一套允许移动程序开发者根据动态的设备性能和散热情况进行调校的新类库集。安卓动态性能框架（ADPF）集成了提供设备热量信息的 Thermal API、帮助安卓系统选择 CPU 的最佳运行点以及核心配置的 Hint API。Unity 的自适应性能包也同时支持这两种 API，使用 Unity 进行开发的团队应该会觉得这很有帮助。

89. Concrete ML

评估

在此之前，我们曾经标记过允许在加密过的数据上进行计算的同态加密技术。Concrete MLl 就是这样一个允许在隐私保护的环境下进行机器学习的开源工具。作为一个基于 Concrete 构建的工具，它帮助数据科学家们简化了完全同态加密（FHE）的使用，帮助他们将机器学习的模型自动转化为同态加密过的数据。此外，Concrete ML 的内置模型中还有和他们的机器学习算法库几乎相同的 API。你也可以通过 Concrete ML 的转换 API 将 PyTorch 网络进行完全同态加密。然而，需要注意的是，若在没有调校过的硬件 中使用 Concrete ML 会导致完全同态加密的速度变慢。

90. Crabviz

评估

Crabviz 是一个用于创建调用图的 Visual Studio Code 插件。这些图表是交互式的，这在使用中等规模的代码库（例如微服务）时起到了很大的作用。它们按文件分组显示类型、方法、函数和接口，并显示函数调用关系和接口实现关系。因为 Crabviz 基于语言服务器协议，只要安装了相应的语言服务器，就可以支持任意数量的语言。虽然这也意味着 Crabviz 仅限于静态代码分析，可能不足以满足某些用例。该插件是用 Rust 编写的，可在 Visual Studio Code Marketplace 上获取。

91. Crux

评估

Crux 是一个用 Rust 编写的开源跨平台应用开发框架。受到 Elm 架构的启发，Crux 将业务逻辑代码作为核心，UI 层则使用了原生框架（如 SwiftUI、Jetpack Compose 以及 React/Vue），或者是基于 WebAssembly 的框架（如 Yew）。通过 Crux，你可以在 Rust 中编写无副作用的行为代码，并在 iOS、Android 和网页中共享。

92. Databricks Asset Bundles

评估

Databricks 最近发布了 Databricks Asset Bundles（DABs）的 公开预览版，它包含在 Databricks CLI 0.205 及更高版本中，正成为打包 Databricks 资源进行源代码控制、测试和部署的官方推荐方式。DABs 在我们的团队中逐渐取代了 dbx。DABs 支持将工作流、作业和任务的配置以及要在这些任务中执行的代码打包成一个可以部署到多个环境的捆绑包。它附带了常见资产类型的模板，并支持自定义模板。虽然 DABs 包含笔记本模板并支持将它们部署到生产环境，但我们仍然建议不要将 笔记本用于生产环境，而是鼓励有意地编写生产代码，并采用支持此类工作负载的可维护性、弹性和可扩展性的工程实践。

93. Electric

评估

Electric 是一种用于移动和 Web 应用程序的本地优先同步框架。本地优先是一种开发范式，其中应用程序代码直接与嵌入式本地数据库对话，并通过双活数据库复制到中央数据库，在后台进行数据同步。使用 Electric，您可以将 SQLite 作为本地嵌入式选项，并将 PostgreSQL 作为中央存储。尽管本地优先极大地改善了用户体验，但它并非没有挑战，CRDT 的发明者们已经在 Electric 框架上努力解决这些问题，以减轻困扰。

94. LiteLLM

评估

LiteLLM 是一个库，通过 OpenAI API 格式 的标准化交互实现与各种大语言模型（LLM）提供商的 API 的无缝集成。它广泛支持各类提供商和模型，并具备一个用于完成、嵌入和图像生成功能的统一界面。LiteLLM 通过将输入转换为匹配每个提供商特定端点要求的方式，简化了集成。在当前环境下，这特别有价值，因为缺乏标准化的 LLM 提供商 API 规范会导致项目中包含多个 LLM。我们的团队已经利用 LiteLLM 在 LLM 应用中更换底层模型，解决了一个重大的集成挑战。然而，需要认识到，它对相同提示的模型有不同响应，这表明仅仅一致的调用方法可能不足以完全优化完成性能。请注意，LiteLLM 还有其他一些特性，如 代理服务器，这些不在本次讨论范围内。

95. LLaMA-Factory

评估

我们一如既往地提醒大家，非必要情况下，不要着急对大语言模型进行微调 —— 这将增加显著的成本和专家资源负担。在必须微调的情况下，我们推荐 LLaMA-Factory 。它是一个开源的、易于使用的 LLMs 微调和训练框架。支持 LLaMA、BLOOM、Mistral、Baichuan、Qwen 和 ChatGLM，它使微调等复杂概念相对容易理解。我们的团队成功地使用了 LLaMA-Factory 的 LoRA 调优来训练 LLaMA 7B 模型。如果您需要进行微调，这个框架是值得评估的。

96. MLX

评估

MLX 是一个开源的数组框架，专为在苹果芯片上进行高效灵活的机器学习而设计。它使得数据科学家和机器学习工程师可以访问集成的 GPU，并在这个基础上选择最适合其需求的硬件。MLX 的设计灵感来自于诸如 NumPy、PyTorch 和 Jax 这样的框架。MLX 的特殊之处在于统一内存模型，它消除了 CPU 和 GPU 之间数据传输的成本，从而实现更高的执行速度。这个特性使得在诸如 iPhone 等设备上运行模型成为可能，为设备端的 AI 应用开辟了巨大的机会空间。尽管这是一个小众领域，但这个框架值得机器学习开发者社区尝试。

97. Mojo

评估

Mojo 是一种新的以人工智能为先的编程语言。它旨在通过将 Python 语法和生态系统与系统编程和元编程特性相结合，缩小研究和生产之间的差距。它是第一种利用新的 MLIR 编译器后端的语言，拥有零成本抽象、自动调优、及早销毁、尾调用优化和更好的单指令 / 多数据（SIMD）工学等炫酷功能。我们非常喜欢 Mojo，并鼓励您尝试一下。Mojo SDK 目前适用于 Ubuntu 和 macOS 操作系统。

98. Otter

评估

Otter 是一个在 Go 语言中非竞争式的缓存库。虽然 Go 语言有几个类似的库，但我们想要强调 Otter 有两个原因：它具有出色的吞吐量并且能够巧妙地实现 S3-FIFO 算法，以获得良好的缓存命中率。Otter 还支持泛型，因此用户可以将任何可比较的类型用作键，并将任何类型用作值。

99. Pkl

评估

Pkl 是最初在苹果公司内部使用、现在已开源的配置语言和工具。Pkl 的关键特性是其类型和验证系统，允许在部署之前捕获配置错误。它生成 JSON、.plist、YAML 和 .properties 文件，并拥有广泛的 IDE 和语言集成，包括代码生成。

100. Rust for UI

评估

Rust 的影响力与日俱增，很多最近出现的构建工具和命令行工具都是由 Rust 编写的。我们现在观察到 Rust 有移植到 UI 开发领域的趋势。大部分偏向于在前后端使用统一代码语言的开发团队会选择 JavaScript 或者 TypeScript 作为后端语言。然而，开发者现在也可以通过 WebAssembly 在浏览器里使用 Rust 代码，而这正在变得越来越常见。另外，像 Leptos 和 sauron 这样的框架会更关注在网页应用开发上，同时 Dioxus 还有其他的一些框架在网页应用开发之外还提供了跨平台桌面以及移动应用开发的支持。

101. vLLM

评估

vLLM 是一个具有高吞吐量和高效内存的大语言模型（LLM）推理和服务引擎，其特别有效的原因在于它可以对传入请求进行连续批处理。它支持几种部署选项，包括使用 Ray 运行时进行分布式张量并行推理和服务部署，在云中使用 SkyPilot、NVIDIA Triton、Docker 和 LangChain 进行部署。我们团队的经验是在本地虚拟机中运行基于 docker 的 vLLM worker，集成了与 OpenAI 兼容的 API 服务器，并在此基础上被一系列应用所利用（包括用于编码辅助以及聊天机器人的 IDE 插件）。团队对此反馈良好。我们的团队利用 vLLM 运行诸如 CodeLlama 70B、CodeLlama 7B 和 Mixtral 等模型。引擎的另一个显著特点是其可扩展能力：只需进行一些配置更改，就可以从运行 7B 模型转换为 70B 模型。如果您希望将 LLMs 投入生产，那么 vLLM 值得进一步探索。

102. Voyager

评估

Voyager 是为 Android 的 Jetpack Compose 构建的导航库。它支持多种导航类型，包括线性、底部表单、标签和嵌套，其屏幕模型与流行框架如 Koin 和 Hilt 集成。在多平台项目中使用 Jetpack Compose 时，Voyager 是实现跨所有支持平台的常见导航模式的优秀选择。近期，Voyager 的开发又重新活跃起来，并在 2023 年 12 月交付了 1.0 版本。

103. WGPU

评估

wgpu 是一个基于 WebGPU API 的 Rust 图形库，它以能够高效处理 GPU 上通用图形和计算任务的能力而闻名。wgpu 旨在填补由于淘汰旧的图形标准（如 OpenGL 和 WebGL）而留下的空白。它引入了一种现代的图形开发方法，既适用于本地应用程序，也适用于基于 Web 的项目。它与 WebAssembly 的集成进一步使得图形和计算应用程序能够在浏览器中运行。wgpu 的出现是一种进步，使高级图形编程对 Web 开发人员更加可访问，应用范围更广泛，从游戏到创建复杂的 Web 动画，将 wgpu 定位为一种令人振奋的技术。

104. Zig

评估

Zig 是一种新的编程语言，它与 C 语言有许多共同点，但具有更强的类型系统、更容易的内存分配以及对命名空间的支持等，此外还有许多其他特性。Zig 的目标是提供一个非常简单的语言，具有直接明了的编译过程，最小化副作用，并提供可预测、易于追踪的执行。Zig 还简化了访问 LLVM 的跨平台编译能力。我们的一些开发者发现这个特性非常有价值，他们甚至在不编写 Zig 代码的情况下，也能使用 Zig 作为交叉编译器。我们看到行业中的团队使用 Zig 帮助构建 C/C++ 工具链。对于那些正在考虑或已经使用 C 的应用程序，Zig 是一种值得探索的新颖语言。

105. LangChain

暂缓

在上一期 Radar 中，我们提到了一些关于 LangChain 出现的批评言论。自那以后，我们愈发对其充满警惕。虽然这个框架为构建大语言模型应用提供了一套强大的功能，但我们发现它使用起来很困难且过于复杂。LangChain 在这个领域早期获得了人气和注意力，这使得它成为了许多人的默认选择。然而，随着 LangChain 试图发展并快速跟进最新变化，开发者越来越难以跟上这些概念和模式的变更。我们还发现其存在 API 设计不一致且冗长的情况。因此，它经常会掩盖底层实际发生的情况，使得开发者难以理解和控制 LLMs 及其周围的各种模式在背后实际是如何工作的。我们将 LangChain 移动到了“暂缓”环，以反映这一点。在我们的许多用例中，我们发现使用更轻量的的专门框架进行实现就足够了。根据用例，你还可以考虑其他框架，如 Semantic Kernel、Haystack 或 LiteLLM。

想要了解技术雷达最新的新闻和洞见?

点击订阅，以接收每两个月一次、来自 Thoughtworks 的技术洞察和未来趋势探索邮件。

现在订阅

Thoughtworks 是一家全球性软件及技术咨询公司，集战略、设计和工程技术咨询服务于一体，致力于推动数字创新。我们在 19 个国家 / 地区的 48 个办公室拥有超过 10,500 名员工。 在过去的 30 年里，我们为全球各地的众多合作伙伴倾力服务，与客户一起创造了非凡的影响力，帮助他们以技术为优势解决复杂的业务问题。

/thoughtworks

Strategy. Design. Engineering.